Usando sus Datos

Cómo se usan los datos, una vez estén cargados y asociados a los elementos del DOM? A continuación se muestra el código de la sesión anterior.

```
var dataset = [ 5, 10, 15, 20, 25 ];

d3.select("body").selectAll("p")
        .data(dataset)
        .enter()
        .append("p")
        .text("New paragraph!");
```

Si se cambia la última línea a:

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
.text(function(d) { return d; });
```

Se puede ver cómo el código cambia la página de ejemplo.

Vean! Utilizamos los datos para actualizar el contenido de cáda párrafo, todo gracias a la magia del método `data()`. Cuando se encadenan métodos, es posible en cualquier momento invocar el método `data()` y crear una función anónima que acepta `d` como parámetro de entrada. El método mágico `data()` asegura que `d` obtenga el valor correspodiente al conjunto de datos original, para el elemento actual.

El valor del "elemento actual" cambia cada vez que D3 hace un recorrido por los elementos. Por ejemplo, cuando se recorre por tercera vez, el código del ejemplo crea la tercera etiqueta `p` y `d` entonces asume el tercer valor del conjunto de datos (`dataset[2]`). Por consiguiente, el tercer párrafo queda actualizado con el valor de texto "15".

Funcionando

En caso de que usted esté empezando a escribir sus propias funciones (o métodos), la estructura básica de la definición de una función es la siguiente:

```
function(input_value) {
    //Calculate something here
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
        return output_value;
}
```

Esta función es muy sencilla, básicamente:

```
function(d) {
        return d;
}
```

y está encapsulada dentro de la función `text()` de D3, por consiguiente lo que devuelva esta función será pasado a la función `text()`.

```
.text(function(d) {
        return d;
});
```

Pero se puede crear algo más sofisticado, porque estas funciones se pueden adecuar de muchas maneras. Es verdad que escribir código en JavaScript es placentero y doloroso a la vez. El punto importante es saber que se pueden definir funciones para lo que se necesite. Probablemente, le gustaría añadir algún texto, el cual produce el siguiente

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

resultado

```
    .text(function(d) {
        return "I can count up to " + d;
    });
```

A los datos les gusta tener casa

Si se está preguntando por qué toca escribir `function(d)...` en vez de `d`, miremos este ejemplo el cual no funciona:

```
.text("Puedo contar hasta " + d);
```

En este contexto, si no se encapsula `d` dentro de una función anónima, `d` no tiene valor. Piense en `d` como un valor temporal que está solo y necesita el lugar caluroso que le proporcionan los paréntesis de la función.

Acá, la función anónima se encarga de darle el espacio que necesita `d`.

```
    .text(function(d) {  //
        return "Puedo contar hasta " + d;
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    });
```

La razón para usar esta sintaxis es que `.text(), attr()`, y muchos otros métodos de D3 reciben una función como un argumento. Por ejemplo, `text()` puede recibir o una cadena de caracteres como un argumento:

```
.text("Alguna cadena")
```

O los resultados de una función:

```
.text(algunaFuncion())
```

o las función anónima en sí puede ser un argumento, como cuando se escribe:

```
.text(function(d) {
        return d;
})
```

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Arriba, se está definiendo una función anónima. Si D3 encuentra una función acá, la procesa y le pasa el valor actual de `d` como el argumento de la función. Si no existe una función, D3 no puede saber a quién pasarle el argumento `d`.

Inicialmente esto parece no tener sentido y que es trabajo adicional solamente para obtener el valor de `d`, pero próximamente este enfoque será más claro a medida que miremos temas más complejos.

Más allá del Texto

Las cosas se vuelven aún más interesantes cuando empezamos a mirar otros métodos de D3, como son `attr()` y `style()`, los cuales permiten actualizar respectivamente los atributos de HTML y propiedades de CSS sobre selecciones.

Por ejemplo, con una sola línea de código adicional, se produce el siguiente resultado.

```
.style("color", "red");
```

Ahora todo el texto es de color rojo. Lo interesante es que se puede utilizar una función para que el texto se despliegue en rojo solamente cuando el dato está por encima de cierto umbral. Entonces, es posible alterar la última línea incertando la función:

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    .style("color", function(d) {
            if (d > 15) {   //Umbral - 15
                return "red";
            } else {
            return "black";
            }
        });
```

Resultados. Puede ver que las primeras tres líneas están en negro, pero apenas `d` excede el número 15, el texto cambia de color a rojo.

En la siguiente lección, se utilizará `attr()` y `style()` para manipular `divs`, y generar un diagrama de barras simple - nuestra primera visualización!

Siguiente: Desplegar Divs–>

Este tutorial fue traducido por Gabriel Coch
Todo el contenido fue desarrollado por y le pertenece a Scott Murray

open in browser PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com